Mission Control — Task Management System

Mission Control — Complete Reference

> Last Verified: 2026-02-17 | Owner: John

Mission Control (MC) is the active task management system. It replaced Taskwarrior and enforces GOTCHA compliance via hooks.

Overview

Status: PRIMARY task system (replaced Taskwarrior) **Backend:** SQLite (`~/system/databases/mission-control.db`) **Frontend:** Web dashboard at http://localhost:3030 **Enforcement:** gotcha-enforcer.py reads `/tmp/mc-active-task` + DB **Daemons:** com.john.mc-dashboard + com.john.mc-session-worker

---

CLI Commands

Basic Operations

```
# List all open tasks
node ~/system/tools/mc.js list

# List my tasks only
node ~/system/tools/mc.js list --owner john
```

```
# Add new task
node ~/system/tools/mc.js add "Title" --desc "Description" --priority H --owner john

# Start task (unlocks Write/Edit)
node ~/system/tools/mc.js start <id>

# Complete task with outcome summary
node ~/system/tools/mc.js done <id> "outcome"

# Pause task (blocks Write/Edit)
node ~/system/tools/mc.js pause <id>

# Resume paused task
node ~/system/tools/mc.js resume <id>

# Block task with reason
node ~/system/tools/mc.js block <id> "reason"

# Assign to owner
node ~/system/tools/mc.js assign <id> <owner>

# Show full details
node ~/system/tools/mc.js show <id>

# Show audit trail
node ~/system/tools/mc.js history <id>

# Who's working on what
node ~/system/tools/mc.js active

# Summary counts
node ~/system/tools/mc.js stats
```

Backward Compatibility

```
# Old task.sh wrapper (proxies to mc.js)
~/system/tools/task.sh list
~/system/tools/task.sh add "Title"
```

```
~/system/tools/task.sh start <id>
~/system/tools/task.sh done <id>
```

Dashboard (CEO UI)

URL: http://localhost:3030 **Features:**

- CRUD operations
- Pause/resume tasks
- Assign ownership
- Priority updates
- Mobile-friendly
- Auto-refresh
- LAN accessible

LaunchAgent: com.john.mc-dashboard (auto-start on boot)

Enforcement — GOTCHA Integration

How It Works

1. **Start task:** `mc.js start <id>` creates `/tmp/mc-active-task` with task ID
2. **Hook reads:** `gotcha-enforcer.py` checks `/tmp/mc-active-task` on every Write/Edit/Bash
3. **Validates:** Hook queries `mission-control.db` to verify task exists and is active
4. **Blocks:** If no active task → BLOCKED. Must `mc.js start` first.
5. **Done:** `mc.js done` removes `/tmp/mc-active-task`, blocks further edits until next start

GOTCHA Checklist Requirement

Rule: BEFORE any Write/Edit/Bash, you must have `/tmp/gotcha-task-{id}.md` with 6 sections:

```
# GOTCHA Checklist — MC Task #{id}

## G — Goal
Šta tačno radim? Koji spec/kriterij?

## O — Options
Koje opcije imam? Koju biram i zašto?
```

```
## T — Tools
Koji alati? Jesam li provjerio manifest?

## C — Context
Šta sam pročitao/verificirao prije?

## H — Hazards
Šta može poći po krivu? Backup plan?

## A — Acceptance
Kako ću znati da je gotovo?
```

Enforcement: Hook blocks Write/Edit until checklist exists.

Rules

1. **Svaki task od Alema → ODMAH u MC**
   - `mc.js add "Title" --desc "X" --priority H --owner john`
2. **Prije rada →** `mc.js start <id>`
   - Kreira `/tmp/mc-active-task`
   - Unlocks Write/Edit/Bash
3. **Kad završiš →** `mc.js done <id> "outcome"`
   - Removes active task flag
   - Blocks further edits
   - Records outcome in DB
4. **Kraj sesije →** `mc.js list`
   - Check remaining open tasks
   - Create follow-up tasks if needed

Session Execution Tools

These are ephemeral — die when session ends:

```
// Create task (visible in Claude Code UI only)
TaskCreate({
  subject: "Task title",
  description: "Full description",
```

```
  activeForm: "focus" | "backlog"
})

// Update task status
TaskUpdate({
  taskId: "task_xxx",
  status: "completed" | "cancelled"
})
```

Use cases:

- Quick sub-tasks during implementation
- Temporary tracking within session
- NOT for persistence — use MC for that

Database Schema

File: `~/system/databases/mission-control.db`

Tables:

- `tasks` — Core task data (id, title, description, status, priority, owner, created_at, updated_at)
- `task_history` — Audit trail (task_id, action, actor, timestamp, metadata)
- `task_assignments` — Assignment tracking

Status values: `open`, `active`, `paused`, `blocked`, `done`, `cancelled`

Priority values: `L` (low), `M` (medium), `H` (high)

Owner values: `john`, `alem`, `edita`, `—` (unassigned)

Integration with Other Systems

Event Bus

MC emits events on task changes:

- `task.created` — New task added

- `task.status_changed` — Status updated
- `task.assigned` — Owner changed
- `task.completed` — Task marked done

Subscribers: pipeline-watcher, autowork daemon, session-worker

HiveMind

Completed tasks logged to HiveMind:

```
node ~/system/agents/hivemind/hivemind.js post john success "MC #<id>: <outcome>"
```

Migration from Taskwarrior

Status: Taskwarrior still installed but NOT primary **Source of truth:** MC DB **Old commands:** `task list` → use `mc.js list` **Data:** Not migrated. MC started fresh.

For CLI implementation, see `~/system/tools/mc.js` (2000+ lines, SQLite + better-sqlite3)

Revision #3
Created 2026-02-17 22:28:47 UTC by John
Updated 2026-05-31 20:01:15 UTC by John